第7章 PID制御の実験

── 内蔵A-D/D-Aコンバータを活用して自動制御を実現する

相田泰志

ここでは，本誌付属基板を使ってPID制御の実験を行う．記事の前半でPID制御について簡単に解説を行った後，実際に回路を動作させる．抵抗とコンデンサのみを追加する簡単な回路である．制御の状態はRS-232-C経由で接続したパソコンを用いて観測するので，比較的簡単に実験を行うことができる．

（編集部）

米国Analog Devices社のマイクロコンバータ「ADuC702x」は，ARMマイコンを核とし，高精度A-Dコンバータと D-Aコンバータを装備した，アナログ処理を得意とするマイクロプロセッサです．このADuC7026を使用して，簡単な制御の実験を行いました．

今回は，電源とわずかな部品を外付けするだけで制御の実験ができるコントローラを作成してみました．付属基板に搭載されているADuC7026は，電源リファレンスやA-Dコンバータ，D-Aコンバータが内蔵されているので，複雑な外付け回路を用意しなくても実験できます．

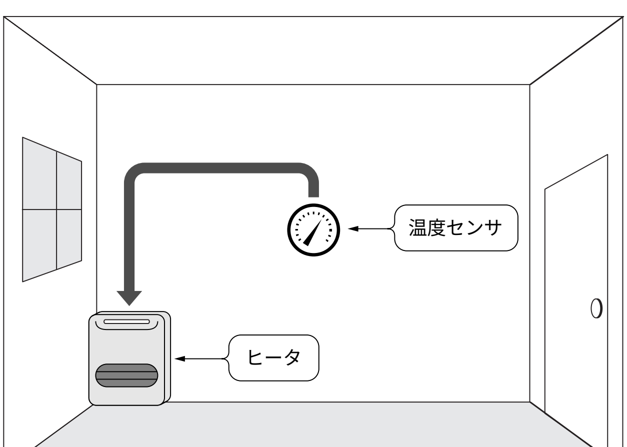


図1　部屋の温度を制御するシステム

温度センサにより部屋の温度を測定して，設定温度より低い場合にはヒータで部屋を暖める．人の出入りや換気のためにドアや窓が開けられると，部屋の温度は急激に変化する．より短い時間で部屋の温度を設定温度に調整し，その温度を保つためには，くふうが必要になる．

1．自動制御について

現在では，いろいろな場面で自動制御が適用されています．温度制御や速度制御のような「連続制御」と，工作機械などのようにあるパターンに従って動作を行う「シーケンス制御」と呼ばれるものがあります．

● 制御のための動作

温度制御などに使われる連続制御の基本は，

①センサなどにより物理量を測定し（例えば温度），

②その測定値に対して判断を行い（温度調節計），

③それに応じて物理量を変化させた出力を出す（ヒータ，ファンなど）．

④その出力結果により物理量が変化し，また，センサで測定する．

といった一連の動作の連続的な繰り返しです．この一連の動作を制御ループと呼び，制御動作の基本になります．制御ループにより，いろいろな物理量の制御を行うわけです．

目標値（設定値）に対していかに物理量を制御できるかということが制御の目的であり，設定値と実際の値を一致させられることが，制御性が良いということになります．

● 目標値に一致させるのは難しい

目標値に実際の値を合わせるだけならば，一見簡単そうに思えます．しかし，実際のシステムで制御性の向上を実現することは簡単ではありません．

図1のように，ヒータにより部屋の温度を一定にするような制御システムを考えてみます．温度が低ければヒータをONして部屋の温度を上げていきます．部屋が設定温度になったときにヒータをOFFすれば，一見よさそうに思えます．しかし，ヒータに余熱があると，それによって室温が上昇してしまいます．少し早めにOFFすればよさそうですが，早すぎるとこんどは設定温度に達しないことになります．また，実際の住宅であれば，人の出入りや換気で扉や窓の開閉があり，部屋の温度が急激に変化することもあります．このような状況で，室温を一定に保ち，変化が起こったときにできるかぎり早く設定温度までもっていくことは，なかなか難しいのです．

このようなさまざまな条件に対して良い制御性を実現するために，いろいろな制御方法が考えられています．

● 代表的な制御方式は３種類

一般に使用されている代表的な制御方法には以下のようなものがあります．

1）ON/OFF制御

設定値をしきい値としてON/OFFを行う方法です．例えば，こたつなどのサーモ・スイッチを使って実現できます．

2）PID制御

もっともよく使用されている制御方式です．P（比例），I（積分），D（微分）により制御を行います．エアコンなどの家電の制御やモータの速度制御，産業用プラントの制御など，あらゆるところで使用されています．

3）ルール形制御

ファジーなどの経験ルールやモデルを作成することにより制御を行う方式です．家電など，比較的制御条件や範囲が限られた部分で使用されています．

今回は，付属基板を使用して，このなかでもっともポピュラなPIDコントローラを作成してみたいと思います．

2．PID制御の基礎

例えば，わたしたちがコップに水をくむときに，どのような動作をしているか考えてみます（**図2**）．最初は蛇口を思い切り開いて水を入れ，いっぱいになりそうになると蛇口をちょっとずつ閉めていきます．そして，最後はちょろちょろと水を出して，目的の量まで水がたまったら蛇口を完全に閉じるのではないでしょうか．これをそのまま制御に持ち込んだのがPID制御です．

● 比例・積分・微分で制御する

PID制御の式を**図3**に示します．PV（process variable）はプロセス値と呼ばれる測定値，SV（set point variable）はPVを設定したい値，つまり設定値（目標値）を示します．PID制御は，このPVとSVより出力を作り出します．

P（比例）成分は，PVとSVの差（偏差）に対してゲインをかけて出力します．PVとSVの差に比例して出力が決まります．コップに水をくむときの例でいうと，水がたまるに従って蛇口をちょっとずつ閉じていく動作です．

I（積分）成分は，偏差を時間で積分していきます．P動作

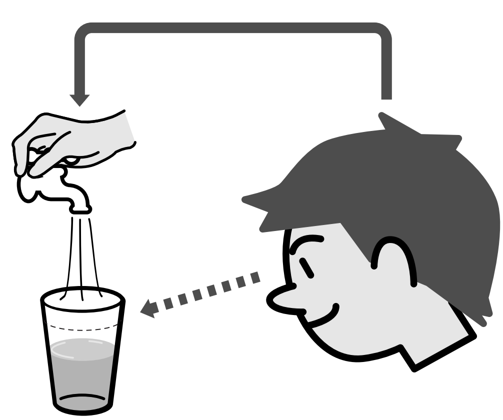


図2　コップに水をくむ動作

最初は蛇口を思い切り開いて水を入れ，いっぱいになりそうになると蛇口をちょっとずつ閉めていく．そして，最後はちょろちょろと水を出して，目的の量まで水がたまったら蛇口を完全に閉じる．

$$e = SV - PV$$

$$MV = K_p \left(\underbrace{e}_{P（比例）} + \underbrace{\frac{1}{T_I}\int e\,dt}_{I（積分）} + \underbrace{T_D \cdot \frac{de}{dt}}_{D（微分）} \right) + MV_0$$

- MV ：操作量
- PV ：プロセス量（測定値）
- SV ：設定値（目標値）
- K_P ：比例ゲイン
- T_I ：積分ゲイン
- T_D ：微分ゲイン
- MV_0 ：MV初期値

図3　PID制御式

PID制御は，測定値と設定値をもとに，比例，積分，微分により出力を決める．

では PV と SV を一致するような出力がありますが，PV と SV の差がわずかになると出力がほとんど出ないことになります．PV がなかなか変化しないといつまでたっても SV に達しません．そこで，偏差が残っている場合に，偏差を積分していき，PV を動かして一致させるものです．

コップに水をくむときの例でいうと，最後にちょろちょろと水を出す動作です．

D（微分）成分は，偏差を微分した値を出力します．これは，SV もしくは PV が変化した場合にすぐに値を近づけるしくみです．コップに水をくむときの例でいうと，最初に蛇口を思い切り開いて水を入れる動作です．室温の制御でいうと，人の出入りでドアが開いたときなどに急に温度が下がり始めたら，すぐに出力を上げるしくみです．

このように，PID制御は，人間の動きに近い動作をするため，わかりやすく，また安全な制御方式として普及しています．

● ディジタルPID制御

前述のPIDの式は，位置形のPID制御と呼ばれます．これをマイコンなどでディジタル制御を行う場合は，時間ごとにサンプリングされるため，現状の PV を PV_n とすると，**図4**の式になります．ただこの式では，積分項が残っています．例えば，出力100％の場合などに測定誤差の関係で偏差が残っていると積分が続けられ，なかなか出力が落ちてこないという問題が発生する場合があります．

そこで，ディジタルPIDでは，それぞれの周期ごとに出力を計算する速度形PID制御が使われます．これは，**図5**のように，MV の偏差を計算する方式です．これだと，出力ごとの制限（出力変化量制限）を加えたり，マニュアルからオートの切り替え時に MV を急激に変化させない（バランスレス・バンプレス制御）などを簡単に実現することができます．

$$e_n = SV_n - PV_n$$

$$MV_n = K_P \left\{ e_n + \frac{\Delta T}{T_I} \sum_{i=1}^{n} \cdot e_i + \frac{T_D}{\Delta T} (e_n - e_{n-1}) \right\} + MV_0$$

ΔT：サンプリング周期

図4　位置系PID制御のアルゴリズム

マイコンなどでディジタル制御を行うときには時間ごとにサンプリングされるため，図3における PV を PV_n とする．

$$\Delta MV_n = MV_n - MV_{n-1}$$

$$= K_P \left\{ (e_n - e_{n-1}) + \frac{\Delta t}{T_I} e_n + \frac{T_D}{\Delta T} (e_n - 2 \cdot e_{n-1} + e_{n-2}) \right\}$$

$$MV_n = MV_{n-1} + \Delta MV$$

図5　速度系PID制御のアルゴリズム

周期ごとに出力を計算する．

● PIDコントローラに求められる機能

コントローラとして動作させるために，いくつかの機能を実装する必要があります．

1）不完全微分

PIDの式は微分で表されています．このため，式をそのまま適用すると，変化があったとき（周期）にしか出力せず，すぐに出力が0になってしまいます．このため，実際にヒータやバルブなどが応答できず，制御に影響を与えないことになります．そこで，PIDコントローラでは，不完全微分と呼ばれる微分に遅れを持たせた信号が使用されます（**図6**）．

2）フィルタ機能

PIDコントローラで微分（D）動作を用いた場合，アナログ変換器の分解能そのものが D 動作に影響を与えてしまう場合があります．また，偏差がそのまま出力に影響するため，制御に無関係なノイズ成分はできるかぎり取り除く必要があります．ハードウェア的にフィルタなどで除去する方法もありますが，ソフトウェア的に平均値フィルタや移

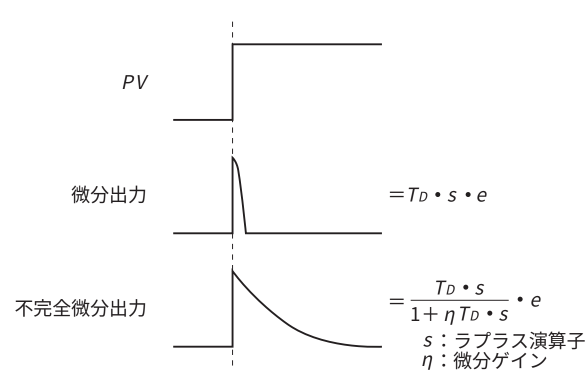


図6　不完全微分

微分に遅れを持たせる．

動平均フィルタ(**図7**)で除去することにより，影響を低減できます．また，移動平均フィルタの場合には，定数により時定数を変えられるので，応答速度やプロセスによって変化させることも可能です．

3）出力変化率制限機能

出力は制御システムによって異なり，バルブやヒータ，モータなど，いろいろなものがあります．PID演算では，制御対象に関係なく演算されるため，使用する出力デバイスによっては実際の出力に反映されないことも考えられます．そこで，対象に合わせて，出力の変化率を制限する機能を設けます．

4）警報機能

PIDコントローラの重要な機能の一つが警報機能です．上限警報や下限警報，変化率警報など，制御システムの異常をすぐに知らせて，それに対して，出力を保持したり止めたりすることが必要になります．

一般には，「上限」，「上上限」，「下限」，「下下限」というように，上限と下限にそれぞれ2段の警報を設置します．1段目でアラーム，2段目で強制停止などを行います．

5）マニュアル機能

PIDコントローラはいつも自動制御ではなく，*MV*を自分で操作できるようにマニュアルで動作させることもあります．オートとマニュアル動作を切り替えることも，速度形PIDでは簡単に行えます．

3．PID制御機能の実装

前置きが長くなってしまいました．では，PID機能をADuC7026に実装してみましょう．

● 実験用ハードウェアの構成

必要なものは，電源とプログラミング用機材，RS-232-Cを実装したパソコンです．また，制御用の負荷として，適当な電解コンデンサと抵抗を用います．

図8に接続回路を示します．D-Aコンバータの出力に，抵抗とコンデンサの負荷を接続し，A-Dコンバータに戻しています．この抵抗とコンデンサが，制御される対象になります．コンデンサと抵抗のみなので，ボード上に直接はんだ付けしています(**写真1**)．

制御の状態は，パソコンを使って確認します(**図9**，**写真2**)．ADuC7026からRS-232-C経由でパソコンにデータを

7

$$PV_n = \alpha \cdot PV_{n-1} + (1-\alpha) \cdot AI_n$$

AI_n：入力データ
PV_n：フィルタ後データ
α　：フィルタ定数(0～1)

図7　移動平均フィルタ
制御に無関係なノイズ成分は，できるかぎり取り除く必要がある．ソフトウェア的にノイズを取り除くフィルタの例である．

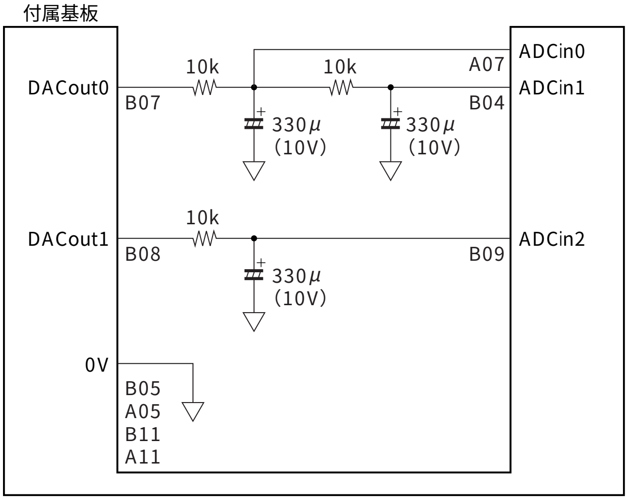


図8　実験用回路
D-Aコンバータの出力に，抵抗とコンデンサの負荷を接続し，A-Dコンバータに戻している．

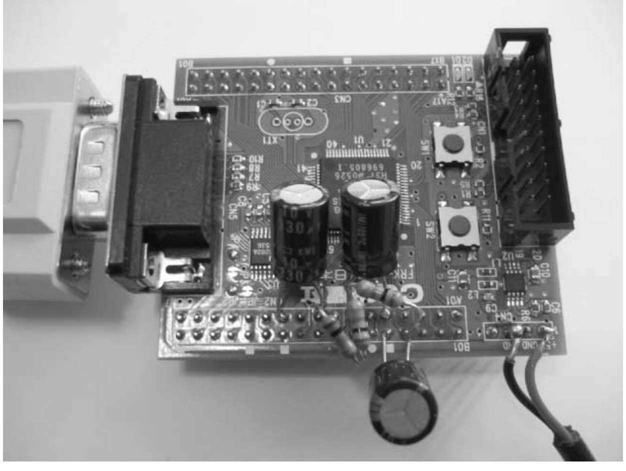


写真1　製作した実験用回路
抵抗とコンデンサはボード上に直接はんだ付けした．

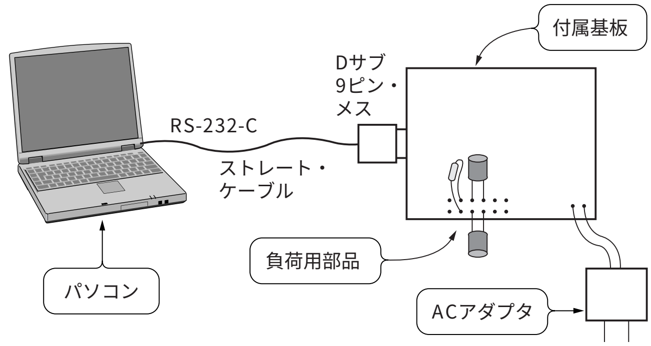


図9　実験システムの構成

制御の状態は，パソコンを使って確認する．ADuC7026からRS-232-C経由でパソコンにデータを送る．

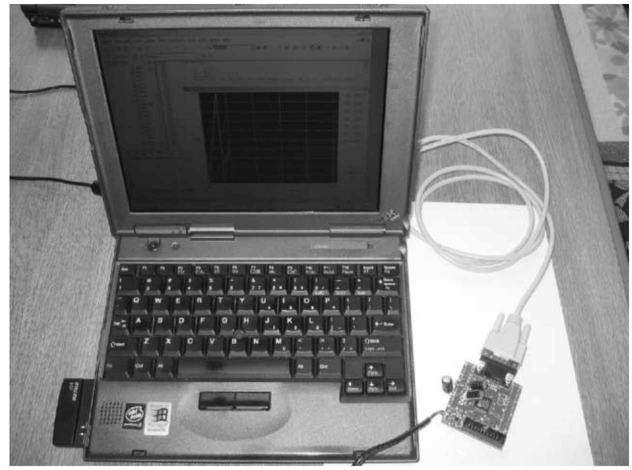


写真2　実験のようす

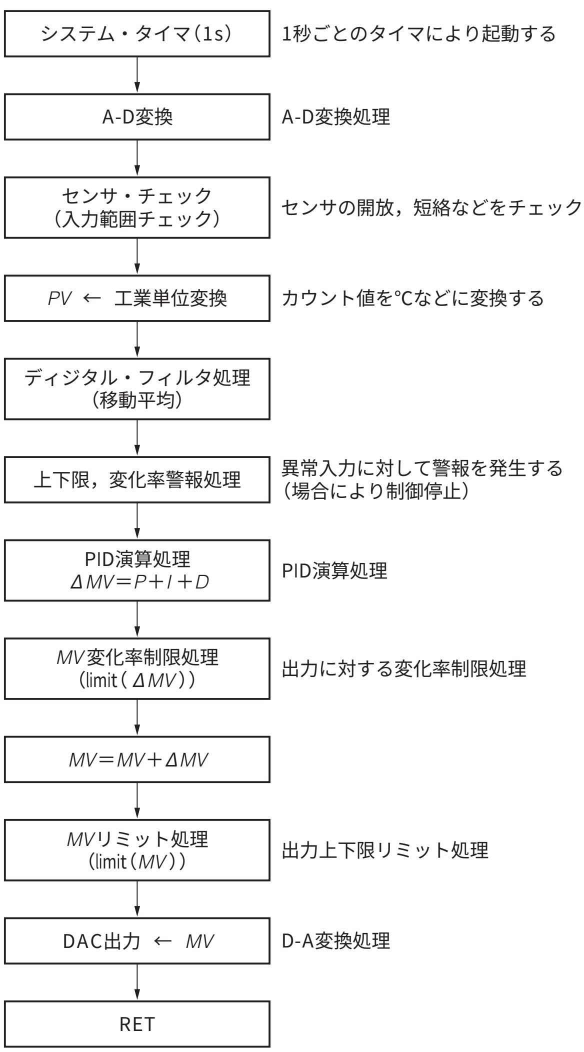


図10　制御ソフトウェアの処理の流れ

リスト1　制御ソフトウェアの演算部

```
v0 = adc(0) / 4096 * 100; // ADC入力  vo= 0-100%

PV = DF * v0 + ( 1 - DF ) * PVOLD; // 移動平均処理

if(PV > PH)  PHALM = 1;        // 上下限入力処理
else if(PV < PH - HYS)  PHALM = 0;

if(PV < PL)  PLALM = 1;
else if(PV > PL + HYS)  PLALM = 0;

P = PV - PVOLD;
I = (SV - PV) / TI;
DMV = KP * ( P + I);            // PI演算処理

PVOLD = PV;

DMV = limit(DMV);               // 変化率制限処理

MV = MV + DMV;                  // ＭＶ値演算処理

if(MV > 100) MV = 100;
if(MV < 0  ) MV = 0;            // MVリミット値セット

vout0 = (MV / 100) * 4096;    //  DACカウント値変換
```

送ります．そしてパソコン側のソフトウェアでグラフィカルに表示させます．具体的には，ADC0～ADC2，DAC0～DAC2の値をグラフで表示します．

● 制御ソフトウェア

制御ソフトウェアの構成を**図10**に示します．システム・クロックに従い，定周期で動作が実行されます．

リスト1に演算部分のコードを示します．今回，カウント値のままで扱っていますが，実際の温度や，速度を扱う場合には，その単位系への変換やPID演算中にオーバフローが発生しないように注意する必要があります．

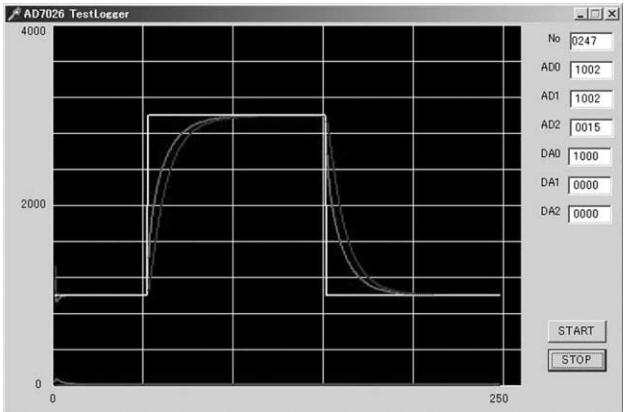


図11　ステップ応答

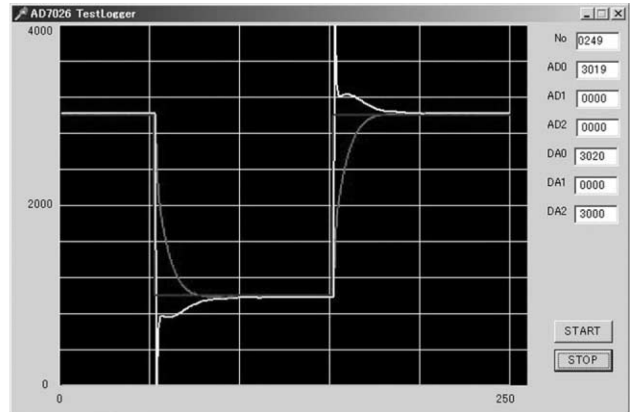
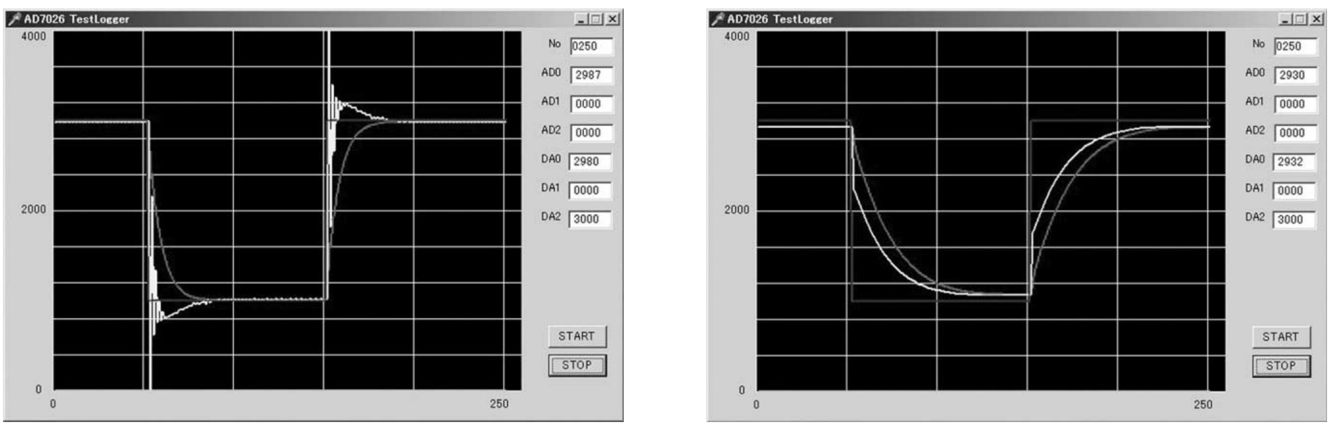


図12　*PI*制御



（**a**）ゲインを大きく　（**b**）ゲインを小さく

図13　比例ゲインを変化させる

7

● ステップ応答の実験

まずは，ステップ信号を出力し，応答を見てみます．

図11のように，25%→75%と出力を変化させます．すると，ADC0に1次遅れの信号が，ADC1にさらに遅れた信号が入っているのがわかります．このステップ応答により，むだな時間と時定数を測定し，PIDの各パラメータを算出することができます．ただしこれは，系の応答がある程度モデルとしてわかっている場合のみです．それ以外の場合には，やはり応答の特性を見ながら，現物合わせでチューニングを行うということが多いようです．

● PI制御の実験

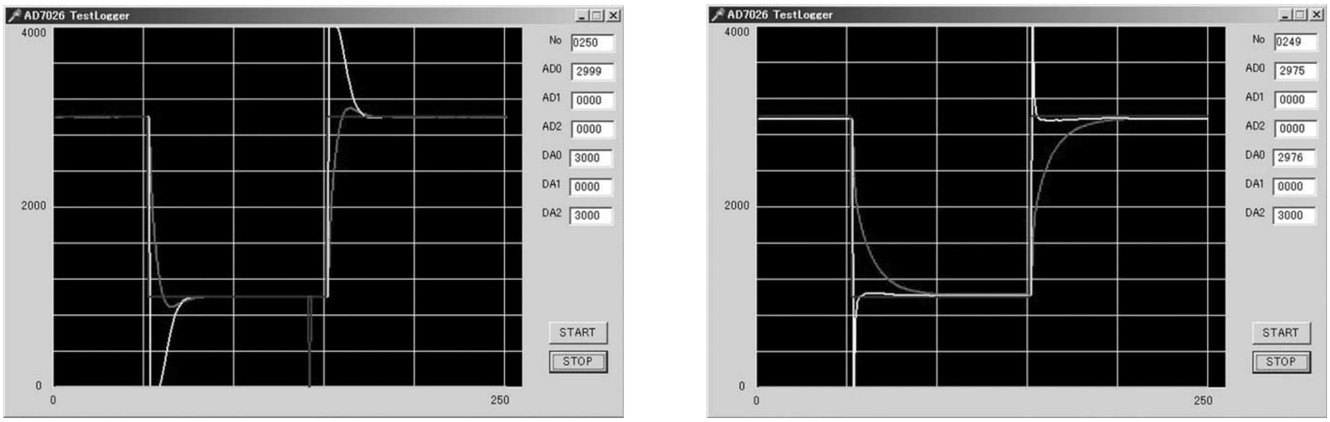
図12にPI制御の結果を示します．微分動作については，系によって使って良い効果がある場合と，外乱などで不安定になってしまう場合があります．ここでは，PI動作のみで制御を行ってみました．

図13にゲインを変えて実験した結果を示します．比例ゲインを大きくすると，だんだんオーバシュートが増え，発振ぎみになっているのがわかります．また，積分ゲインの影響で収束速度が変化するのもわかります（**図14**）．

実際の制御では，このゲインの変更によりチューニングを行います．例えば，早く収束させるのであれば多少オーバシュートぎみに，また，オーバシュートに不つごうのある系ではゲインを下げぎみにしてゆっくり収束させます．

● PID制御の応用

付属基板を使ってPID制御の実験を行いました．今回は，制御対象を外付けの抵抗とコンデンサで行ったため，D-Aコンバータや A-Dコンバータのチャネルにはまだ余裕があります．これらを活用してダミー・プラントを作成するの



（a）ゲインを大きく　　（b）ゲインを小さく

図14　積分ゲインを変化させる

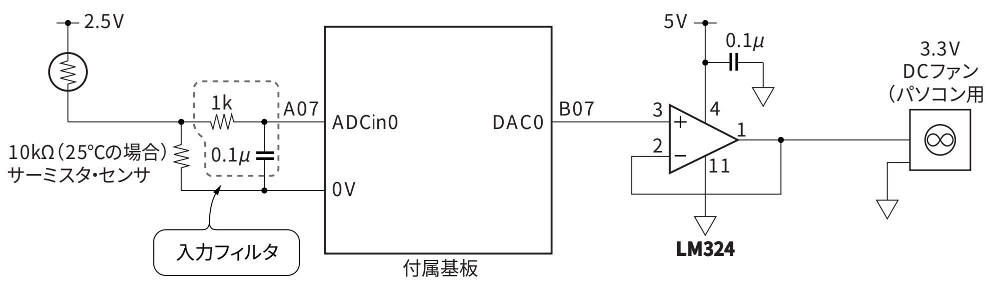


図15
パソコン内部の温度制御の回路例
サーミスタ・センサで温度を測定し，DCファンで温度を制御する．

もおもしろいと思います．もちろん，負荷を変えたり，カスケード制御や比率制御といった，変数が多数あるような制御の試験も簡単に実現できます．

ある程度所望の制御ができるようになれば，実験だけでなく実際に動作させるシステムを作ってみたくなるものです．身近なところでは，パソコン内部の温度制御などを行ってみるのもおもしろいのではないでしょうか．**図15**のようにサーミスタ・センサを取り付け，DCファンを制御することで実現できます．

そうだ・やすし
西山工業（株）